# An Innovation Design of Mobile Platform with Push Message Service on Cloud Environments

Quoc Nguyen

# An Innovative Design of Mobile Framework with Push Message Service in Cloud Environment

Huu-Quoc Nguyen
Computer Engineering Department
Kyung Hee University
Yongin, South Korea
Email: quoc@khu.ac.kr

Tien-Dung Nguyen
Computer Engineering Department
Kyung Hee University
Yongin, South Korea
Email: ntiendung@khu.ac.kr

Eui-Nam Huh
Computer Engineering Department
Kyung Hee University
Yongin, South Korea
Email: johnhuh@khu.ac.kr

*Abstract*—**Currently, mobile services have been increasing rapidly. Mobile devices and applications are built in different platforms (e.g. Android, IOS or Windows Phone, etc.) which required advanced knowledge for developing applications. In addition, a key requirement of pervasive mobile applications is changes occur asynchronously, and it is important that when they occur mobile users are notified in a timely fashion. Without infrastructure for handling the dispatching of asynchronous events, mobile application developers need to code this as part of their applications. This presents developers with extra complexity above and beyond the core application functionality. Most often, the developers need to create again and again the same content delivery mechanism for each application. An alternative is to use a push notification technology that managed event notification. To overcome this issue, this study propose a new design of a mobile framework that supports users to easily adapt to various mobile platforms without having advanced programming knowledge as well as integrating a Push Message Service system on this framework.**

***Keywords—cloud services; mobile programming; Cordova plugin; MEAP; OpenStack***

## I. Introduction

The latest mobile devices and applications are changing the way users communicate, do business and access news and entertainment. Businesses, consumers and programmers have embraced this innovative technology, making mobile application development one of the most demanded and fastest growing IT careers. Mobile developers write programs inside of a mobile development environment using the Objective C, C++, C# or Java programming languages. A mobile application developer chooses the mobile platform they will develop for, such as Googles Android or Apples iOS. However, it takes time to learn the programming languages and software development environment for that platform. As a solution for this issue, we propose a framework that support for the users to easily develop applications could run on many platforms such as Android, IOS, and Windows Phone without requiring advanced programming knowledge. Push Message Service (PMS) [1] helps users in sending event or alert to the application users even when they are not logged into application. There are some predefined types of push notifications such as the tile notification on Windows Phone and a badge notification in iOS, sending custom data are supported on all platforms. Moreover, supporting push notifications at the large scale has been incredibly complicated for mobile application developers. Each popular mobile platform maintains a different free relay service that delivers notifications through persistent connections to devices running the platforms user's own. This means that to support millions of users on multiple mobile platforms, developers must integrate with each of these platform-specific relay services, which introduces operational complexity and cost. That is the reason why we integrate a Push Message Service (PMS) on cloud into this framework. Using Apache Cordova the fastest way we could choose to start building a mobile application for the services which can be run on many platforms. The main objectives of this study are to create an effective environment development as well as bring a push message service to the user.

The rest of this paper is organized as follows: related works are reviewed in section II. Section III describes the system architecture. Section IV presents the Push Message Service. Section V describes the sequence diagram of RmCRC MEAP. The Apache Cordova is shown in section VI. In section VII, we present the results and performance evaluation. Finally, we have summarized our paper and present our future work in section VIII.

## II. Related works

### A. Review of Mobile Platforms

There are various studies which are similar to our approach. In [2], Rapid Interface Builder (RIB) is a browser-based design tool to quickly create prototype and generate the user interface for web applications. The RIB supports UI design by dropping widgets onto a canvas, running the UI in an interactive "Preview Mode" and then exporting the generated HTML5 as well as JavaScript. It also supports jQuery Mobile, Tizen widgets and it runs on the web browsers. Kendo UI HTM5 framework is presented in [3]. Kendo UI mobile applications are built entirely with HTML, JavaScript, and CSS. Kendo UI mobile apps look and feel like the native across mobile platforms. The discussion about Phonegap platform proposed by Mahesh *et al.* [4] uses HTML5, JavaScript, and CSS3 to develop mobile applications. By using PhoneGap, a developer with little or no native language background can start developing mobile applications for all of the popular mobile platforms. In [5], with a C# shared code base, developers can use Xamarin to write native iOS, Android, and Windows apps with native user interfaces and share code across multiple platforms.

However, RIB cannot deploy applications on multiple platforms. It only supports user interaction in web environment. To overcome this issue, our work support users to develop the application in both environments, inside Eclipse IDE and web browser. Both of RIB and Kendo UI does not support the cloud services. Furthermore, cloud computing has become more pervasive now. Thus, supporting the cloud service is essential. In this paper, we also compared the application size, opening time and power consumption between our work and Phonegap, Xamarin platforms.

### B. *Review of Existing Push Message Service*

Today, a modern push notification service need a very broad scope:

- Apple support: iOS push notification, including the new iOS 7 features [6].
- Google support: Google Cloud Messaging push notification, including XMPP notification, upstream notification, and notification synchronization [7].
- Pushover: Pushover makes it easy to get real-time notifications on Android, iPhone, and Desktop. Pushing messages is as easy as using the HTTP libraries available in nearly every programming language [8].
- Microsoft push for Windows Phone [9].
- Kindle Fire is big enough in US and UK that it is now important to support Amazon Push Service [10].

Supporting a large number of push back-ends is more and more critical and limiting to only a small subset of the available features is not an option anymore for developers. However, this means that developers want to be used iOS Push Notification service, they must deploy on iOS platform and Google Cloud Messaging is no exception. So, creating a cross-platform with Push Message Service is a challenge and opportunity.

## III. System Overview

### A. *Overall System Architecture*

Figure 1 illustrates the overall system architecture. It consists of Mobile Enterprise Application Platform layer and Cartridges layer.

*1) Mobile Enterprise Application Platform layer (namely RmCRC MEAP layer):* MEAP software typically comes in the form of mobile middle-ware that connects back-end data sources (enterprise applications and databases) to mobiles devices [11]. It also offers a set of development tools such as HTML/CSS/JavaScript and 4GL rapid application development tools. MEAP tools provide the capability to build and design data models using a graphical editing tool and then translate those elements into applications that can display the content on any type of mobile devices. Mobile applications using MEAPs can be implemented from a central server to mobile devices, regardless of the mobile operating system.

In RmCRC MEAP layer, there is a comprehensive suite of products and services that enable development of mobile applications. It includes services such as VDI, PMS, or mApp Builder, etc. RmCRC MEAP Layer enables users to develop mobile applications without programming background, such that the development of mobile application is affordable, faster time-to-market,

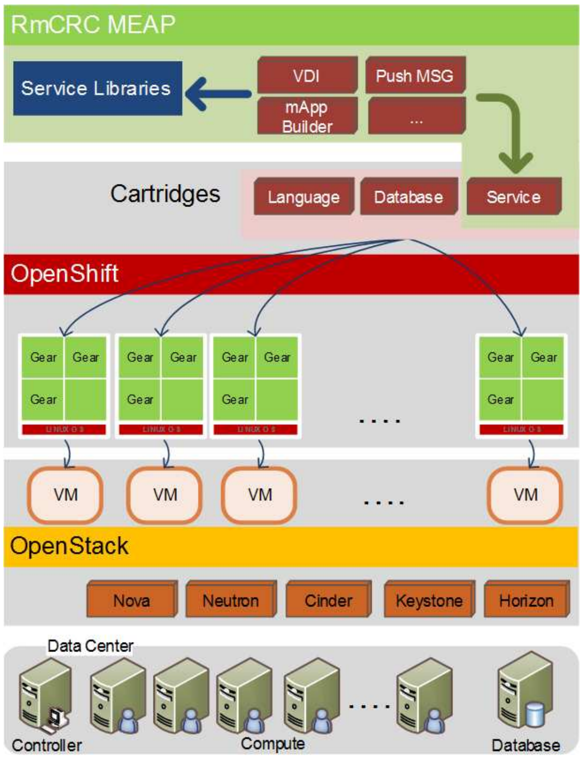


Fig. 1: Overall System Architecture

and employee productivity is improved. Also MEAP management capabilities make it easy to manage devices and applications (maintained in a central location) and installs and updates the mobile software. This layer also supports the developers to connect the Service of Cartridges layer in the application quickly.

*2) Cartridges layer:* Some libraries such as language, database and service installed on the cloud environment. RmCRC MEAP Layer connected to the library component in Cartridges to create a Push Message Service. We will mention about PMS in the next section. In this layer, OpenShift is RedHats cloud development Platform as a Service (PaaS) which allows developers to create, test, run and deploy the applications to the cloud. OpenStack is used to create VMs. There are many free cloud computing platforms, but OpenStack is the optimal choice for this system. The importance of OpenStack, produces a way to minimize or altogether eliminate, type one hyper visor costs because it will run on open-source, free hyper visors. It helps reduce the cost of environment to 40 percent.

### B. *RmCRC MEAP Architecture*

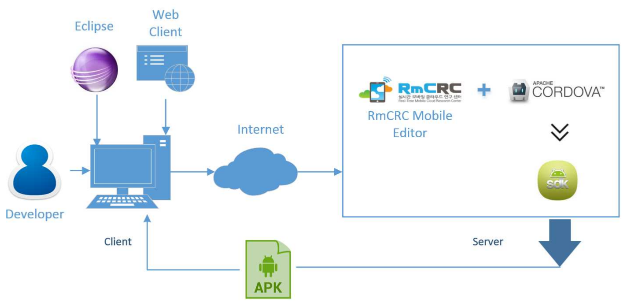


Fig. 2: RmCRC MEAP Architecture

Figure 2 illustrates the RmCRC MEAP architecture consists of two main parts, the server side and the client side.

*1) Server side:* include the RmCRC Mobile Editor, which is a mobile visual tool that we customized and changed to be suitable for our purpose, based on Maqetta open source [12] and the Apache Cordova, which is a set of device APIs that allows a mobile application developer to access native device functions such as the camera or accelerometer through JavaScript.

*2) Client side:* the user could interact with RmCRC Mobile Editor by using web client and plugin in Eclipse IDE. Following steps are suggested: *a) First, create new project and start designing coding on RmCRC Mobile Editor Workspace; b) After that, users start building file by using click Build button on the menu then choose the desired OS platform which the user wants to deploy; c) Finally, receive the execution file from server side.*

## IV. Sequence diagram of RmCRC MEAP

As Figure 4, the sequence diagram of MEAP system is illustrated. The workflow of this progress is represented as follows: *a*) In the beginning, developers start their new project in RMCRC mobile UI editor, which can be run in Eclipse IDE or web browser. Developers then send authentication to the MEAP system and request Service Options such as PMS, etc. After that, MEAP system will create and send a list of requested services to OpenShift in order to manipulate the appropriate VMs from OpenStack which are capable to meet the developers needs. *b*) Under the hood, a couple of steps are declared and run on our server: The target VM and Node is initiated, Gears is installed and managed by OpenShift. After that, Service libraries and project folder structure will be generated and set as default developing environment on Eclipse IDE. *c*) In next step, developers start designing their application via UI and associated HTML, CSS, and jQuery code which are generated in IDE. The developers can request for several cloud services, for example cloud push message for their application until they stop services.

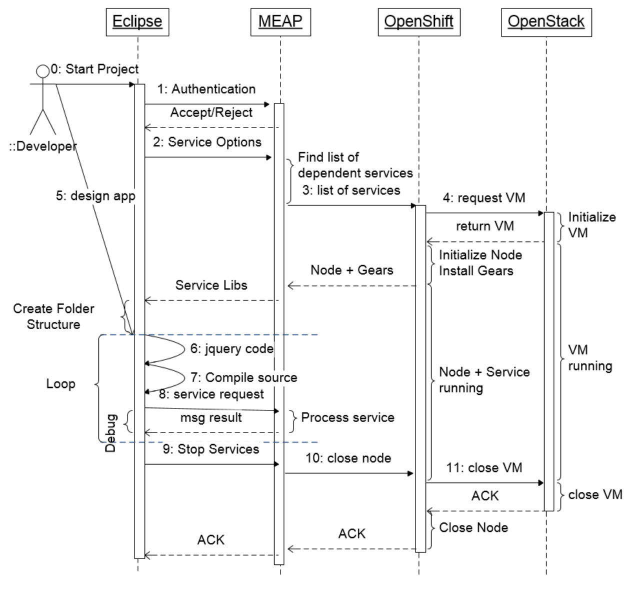


Fig. 4: Sequence diagram of RmCRC MEAP

## V. RmCRC Push message service

RmCRC PMS is a service that allows user to send data from server to users' device, and also to receive messages from devices on the same connection. The PMS service handles all aspects of queuing of messages and delivery to the target application running on the target device.

### *A. Overview of PMS Components*

The detailed main components are presented in the following Table I.

TABLE I: PMS Components

| Component | Description |
|---|---|
| PMS Manager | It is a server provider that will be involved in allowing developers to register, create and get the application ID. One developer can make and manage one or more applications. PMS Manager handles the responsibility of sending notifications from the developer to registered mobile clients as well as between the mobile devices. |
| PMS Agent | It is a mobile agent targeted to each mobile platform (Android, iOS, Windows Phone), which enable cross-platform possibility of this framework. The main task of PMS Agent like a native application always running background on mobile client and communicate with PMS Manager. PMS Agent must be installed before it can be used. |
| PMS API | It is a set of functions such as "send data", "recieve data","send-to-sync", etc. |
| PMS Cordova Plugin | The RmCRC Mobile Editor is mainly by web platform, but the PMS API core is written primarily in C language. That is reason Cordova Plugin is the best choice for this platform. It is an easy way to support the developers can call the APIs are created by multiple languages in web technology. |
| PMS Web Server | PMS Web Server based on PHP. It hepls store Registration ID in the database and handling tasks whenever push notification is needed. |

### *B. PMS Process Flow Diagram*

Figure 3 is illustrating the purpose of each entity involved. Typically, PMS process flow diagram will include the following:

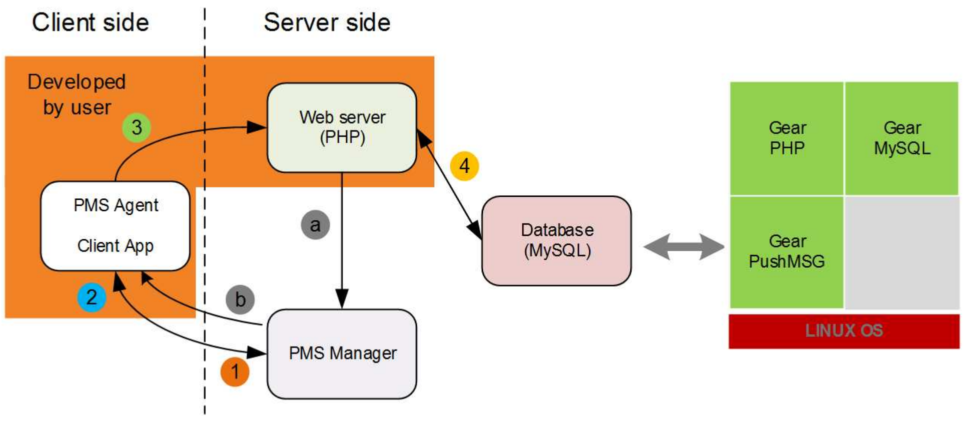


Fig. 3: Push Message Service process steps

*1) First mobile device sends **sender id, application id** to PMS Manager for registration.*

*2) Upon successful registration PMS Manager issues **registration id** to mobile device.*

*3) After receiving registration id, device will send **registration id** to PMS Web Server via PMS Agent.*

*4) PMS Web Server will store **registration id** in the **database** for later usage.*

*a) Whenever push notification is needed, PMS server sends a message to PMS Manager along with device registration id (which is stored earlier in the database).*

*b) PMS Manager will delivers that message to respected mobile device using device registration id.*

PMS Agent must be installed in client application, it is as a mediator between of PMS web server and PMS Manager. In addition, PMS Agent will automatically update the client's IP when network connectivity changes and displays a notification that user has received the new message. In Figure 1, RmCRC MEAP Layer using the service is supported by Cartridges Layer, in this case is Push Message Service. After stored registration id in the database, data will be stored in the VMs on cloud through PushMSG Gear.

## VI. Apache Cordova

Cordova is still packaged as the applications that is using SDKs platform which can be made available for installation from the application store of each device [13]. Combined with a UI framework such as jQuery Mobile, Dojo Mobile, and Sencha Touch, it allows a smart phone application to be developed using only HTML, CSS, and Java Script. Cordova is available for the following platforms: IOS, Android, Blackberry, Windows Phone, and Symbian. Furthermore, these Java Script APIs are consistent across multiple platforms and built on web standard which makes it not portable to other device platforms.

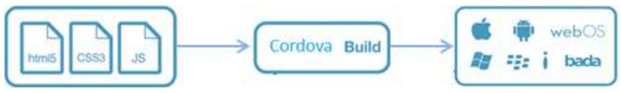


Fig. 5: Cordova compiling process

Figure 5 illustrates the process of a compile web based language in different mobile platforms.

## VII. Result and Performance Evaluation

In Table II, we present a comparison of the features of our proposed system with other systems. We compared RmCRC system with Rapid interface builder, and kendo UI based on mobile UI design, deployment of the application, the integration with Eclipse IDE, connect to third party services, real-time visual code editing. Our system can support the entire components when compared with other systems.

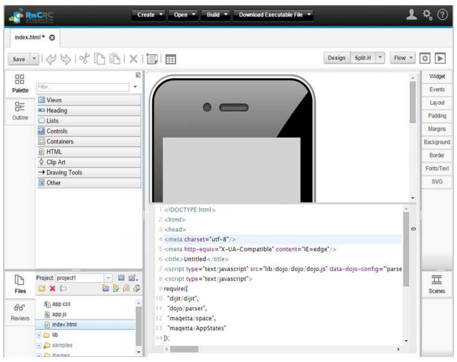
TABLE II: Features comparison of our proposed system with other systems

| | UI Design | Deploy application | Integrate with Eclipse IDE | Connect Services | Code Editing |
|---|---|---|---|---|---|
| RmCRC System | ✓ | ✓ | ✓ | ✓ | ✓ |
| Rapid Interface Builder | ✓ | × | × | × | × |
| Kendo UI | ✓ | ✓ | × | × | ✓ |

TABLE III: Opening time and application size of our system with another system

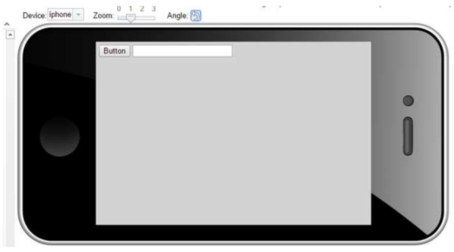
| | Aplication size (KB) | Opening time (ms) |
|---|---|---|
| RmCRC System | 600 | 836 |
| Phonegap | 800 | 920 |

In this paper, we measure the application size and power consumption with and without push message service. We installed the application on SKY Vega Iron A870 (Qualcomm Snapdragon 600 APQ8064T, 2150 mAh of battery power). From Table III, we can derive that the size and the opening time of our application are larger than the Phonegap system for building and running the same "Hello World" application, respectively with the same configuration of mobile devices. There are 300 KB of application files including 200KB of JQuery Mobile, CSS files and 100KB Cordova javascript files. However, in our system, we use a library to display the mobile UI. Thus, the total size is around 600 KB. Phonegap system also does not spend more resources to display the mobile UI, but the application size is 800 KB. Because of the



```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8"/>
<meta http-equiv="X-UA-Compatible" content="IE=edge"/>
<title>Untitled</title>
<script type="text/javascript" src="lib/dojo/dojo/dojo.js" data-dojo-config="'parse
<script type="text/javascript">
require([
  "dijit/dijit",
  "dojo/parser",
  "maqetta/space",
  "maqetta/AppStates"
]);
```

(a) RmCRC Editor Workspace



(b) Preview screen on the browser

Fig. 6: User interfaces of the framework

small application size, access time is faster. As presented in the table III, it takes 800ms to open the application in RmCRC system less than 920ms for the application using Phonegap.

The user interfaces of the platform is also shown in Figure 6. RmCRC Mobile Editor provides WYSIWYG tool which allows the users to perform drag/drop the widgets faster. During the design phase, user can preview the UI design, layout or any arbitray images in preview screen.

TABLE IV: Average power consumption of "Hello World" application without Push Message Service

| Platforms | Power consumption (mW) |
|---|---|
| RmCRC Systems | 81.39 |
| Xamarin | 121.50 |
| Phonegap | 114.0636 |

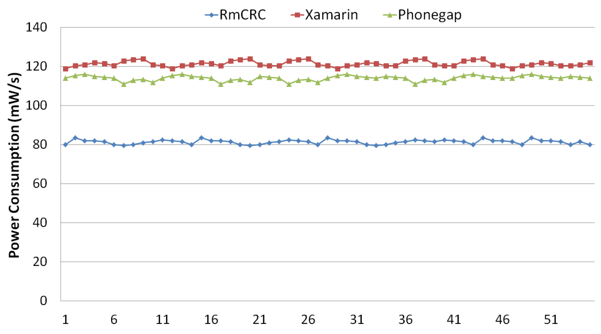


Fig. 7: Power consumption of "Hello World" application

Power consumption of mobile applications has received much attention of the researchers recently [14]. For an effectively use of the mobile's battery, the apps developed by using the cross platform tools should be power efficient. We have measured the power consumption of the app using "Power Tutor" [15]. It is a very popular Android app that reports power consumption of individual apps installed in a mobile device. The result is shown in Table IV and Figure 7. It is to be noted that the reported values are average power consumption of the "Hello World" app using RmCRC Systems, Xamarin and Phonegap platforms. Again, the result points out that the RmCRC's app consumes the least power among the three platforms. The reason is attributed to the fact that the UI is very simple.

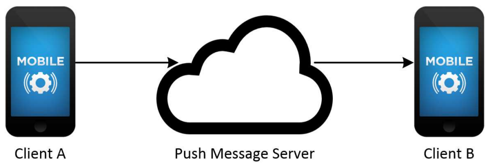


Fig. 8: Testbed System Model

In order to evaluate the power consumption of application with PMS, we have setup a testbed environment to compare RmCRC Push message with a popular mobile push message which is Pushover notification service. The testbed system is shown in Figure 8. We define two testing plans: 3G mode and Wifi mode, we first set the phone to use 3G connection for the first test then use wifi connection for second test. The application started sending messages from client A to client B. After that, we are measuring the percentage of used battery in one hour. The result is shown in Table V.

TABLE V: Message pulling and battery consumption of "Hello World" app with PMS in one hour.

| | 3G | | Wifi | |
|---|---|---|---|---|
| | Pushover | RmCRC | Pushover | RmCRC |
| Battery (%) | 13.15 | 10.2 | 3.6 | 4.5 |
| Messages sent* | 10991 | 30400 | 50514 | 112325 |
| Messages received | 7181 | 27400 | 35500 | 99563 |

*Messages sent as much as possible.

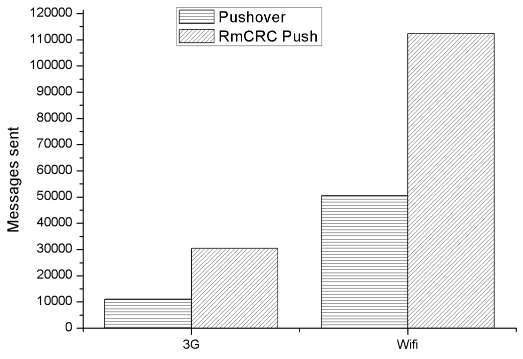


Fig. 9: Messages sent comparison between Pushover and our RmCRC PMS

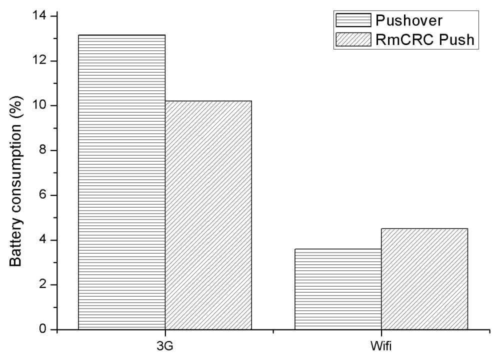


Fig. 10: Battery consumption of two testing plans: 3G mode and Wifi mode

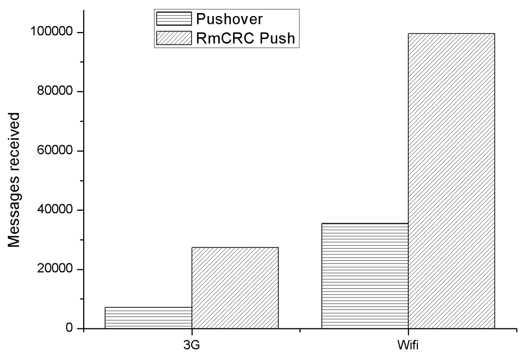


Fig. 11: Messages received comparison between Pushover and our RmCRC PMS

The result shows that by using RmCRC Push to transfer data, we can send and receive a significant number of messages better than traditional HTTP approach of Pushover both in 3G mode and in Wifi mode. Figure 9 represents the speed of messages were received (averaging only 10991 messages per hour for Pushover approach versus 30400 for RmCRC push message approach).

Besides, the experiment showed the reliability of RmCRC PMS, more precisely, the success rate of delivered messages. In 3G mode, using RmCRC PMS all messages are received, whereas HTTP approach in Pushover only handled 7181 of 10991 messages in total, the success rate is 65%. This is caused by the message polling mechanism in traditional HTTP approach. In which, a lot of messages are missed at the interval time between when the connection closed with the previous message and subsequence re-established to receive the next message. This rate is mitigated on Wifi, as the connection can be re-established more quickly. The HTTP can handle 27400 of 30400 messages as seen in Figure 11. In the mechanism of RmCRC PMS, we use the existing connection on the device to send messages. By doing this, our approach can reduce the network latency when sending a specific number of messages to target device.

On the other hand, Figure 10 highlights the efficient in battery consumption of RmCRC PMS. Both in 3G and Wifi mode, RmCRC PMS outperforms HTTP approach in Pushover by manipulating a large amount of the message through the network with less battery consumption.

## VIII. Conclusion and future work

In this paper, we propose an effective platform for mobile application development on cloud environment. This framework helps users to easily develop applications without requiring advanced programming knowledge. As well as, mobile application development framework integrated with cloud services, such as Push Message Service. In future work, we intend to enhance the performance of the system so that it will not only process fast in deployment phase, but also reduce the size of the application.

## References

[1] Na Li, Yanhui Du, Guangxuan Chen, *Survey of Cloud Messaging Push Notification Service*. Information Science and Cloud Computing Companion (ISCC-C), 2013, Guangzhou, pp.273279.

[2] RIB, *Rapid Interface Builder (RIB) Project. Website, 2012*. http://01.org/rapid-interface-builder.

[3] Kendo UI, *Kendo UI HTML5 Framework. Website*. http://www.telerik.com/kendo-ui.

[4] Mahesh Babu R, M. Balaji Kumar, Rakesh Manoharan, M. Somasundaram, S.P.Karthikeyan *Portability of mobile applications*. Software Engineering and Mobile Application Modelling and Development (ICSEMA 2012), International Conference on, Chennai, pp.1-6.

[5] Xamarin, *Xamarin Website* http://xamarin.com/.

[6] Apple, *Apple Push Notification Service.* https://developer.apple.com/library/ios/documentation.

[7] Google, *Google Cloud Messaging for Android.* https://developer.android.com/google/gcm/index.html.

[8] Pushover, *Pushover notifications.* https://pushover.net/.

[9] Microsoft, *Push notifications for Windows Phone.* https://msdn.microsoft.com/en-us/enus/library/windows/apps/ff402558(v=vs.105).aspx.

[10] Amazon, *Amazon SNS - Mobile Push Messaging.* http://aws.amazon.com/sns/.

[11] Wikipedia, *Mobile enterprise application platform MEAP*. http://en.wikipedia.org/wiki/Mobile_enterprise_application_platform.

[12] Maqetta, *Maqetta Open source. Website*. http://maqetta.org.

[13] Apache Cordova, *Apache Cordova. Website, 2012*. http://cordova.apache.org.

[14] Datta, S.K.; Bonnet, C.; Nikaein, N, *Android power management Current and future trends*. First IEEE Workshop on Enabling Technologies for Smartphone and Internet of Things (ETSIoT), 18 June 2012, pp.48,53.

[15] PowerTutor, *PowerTutor App.* http://powerTutor.org.